Introduction in Spring MVC framework

Environment preparation document

by Simion Laurentiu

This document is intended to create a web application useful for java web developers. In order to be able to follow this documentation, some minimal java and eclipse knowledge are required. Tools and technologies used are:

- java 8

- eclipse Oxygen

- tomcat 9

- spring 4

- hibernate 5

- maven 3

- postgresql 9

- JSP

- JSTL

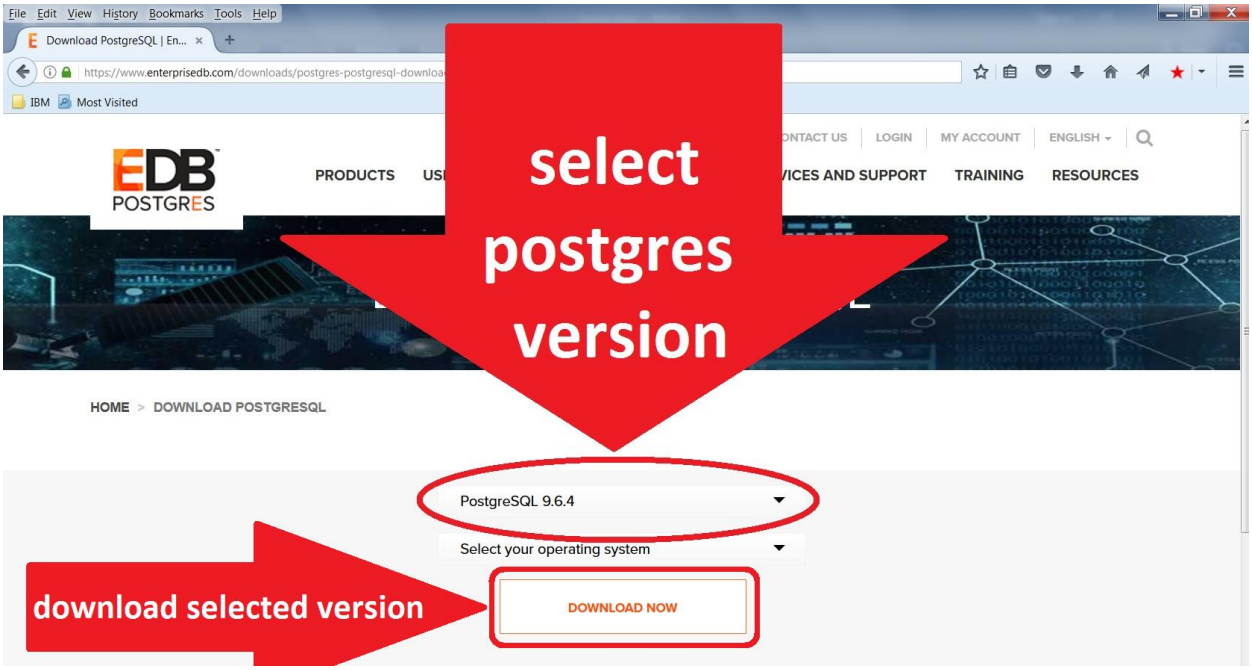
- javascript/jQuery

In order to create and run a spring MVC application, there are some steps of environment preparation. Please follow the next steps:

1. Postgresql

Download and install **Postgresql 9.6.4** from this address:

https://www.enterprisedb.com/downloads/postgres-postgresql-downloads#windows



Try to remember **user** and **password** that you settled. Later, you will set this credentials into database connection:

dataSource.setUrl("jdbc:postgresql://localhost:5432");

dataSource.setUsername("your username");

dataSource.setPassword("your password");

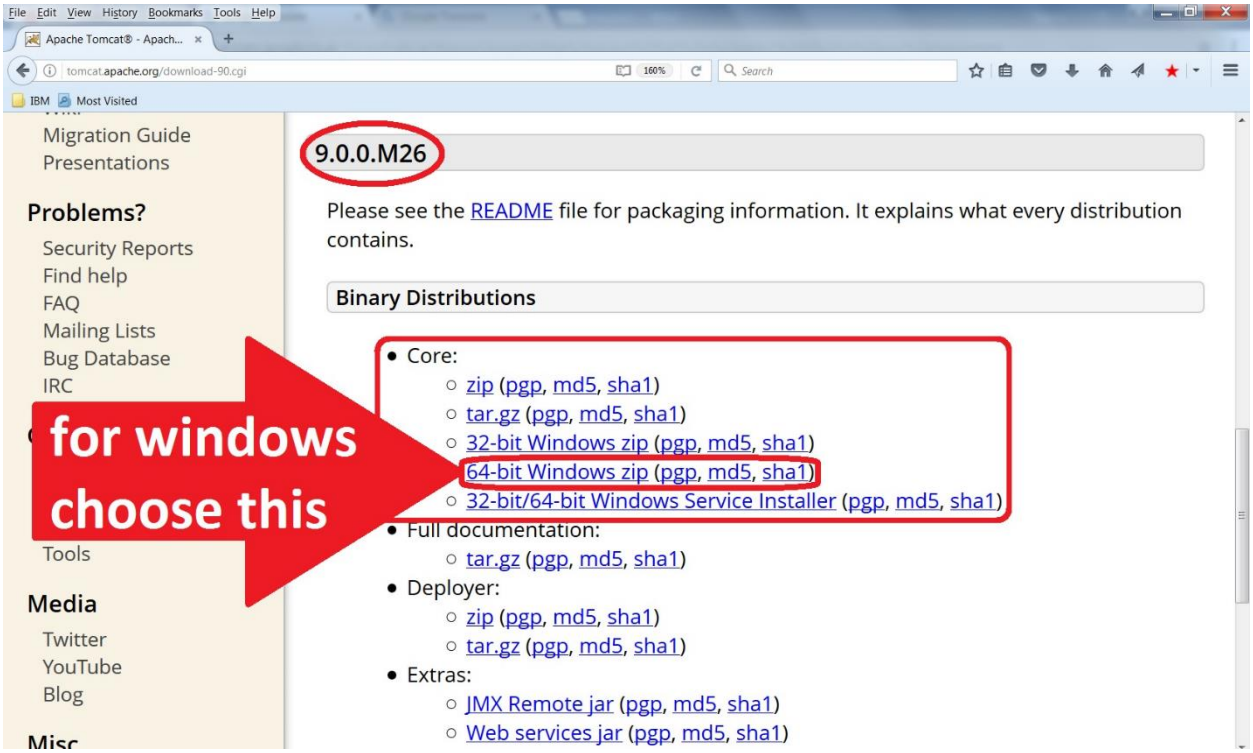
At the end of postgres installation, at last "finish" page, you will have the possibility to install additional features. For our lesson purpose is useless to install additional features/plugins, but if you want you can do it.

2. Tomcat

Download Tomcat 9.0.0.M26 from this address:

http://tomcat.apache.org/download-90.cgi

Scroll down at **9.0.0.M26** and choose the version that is perfect fit for your environment:

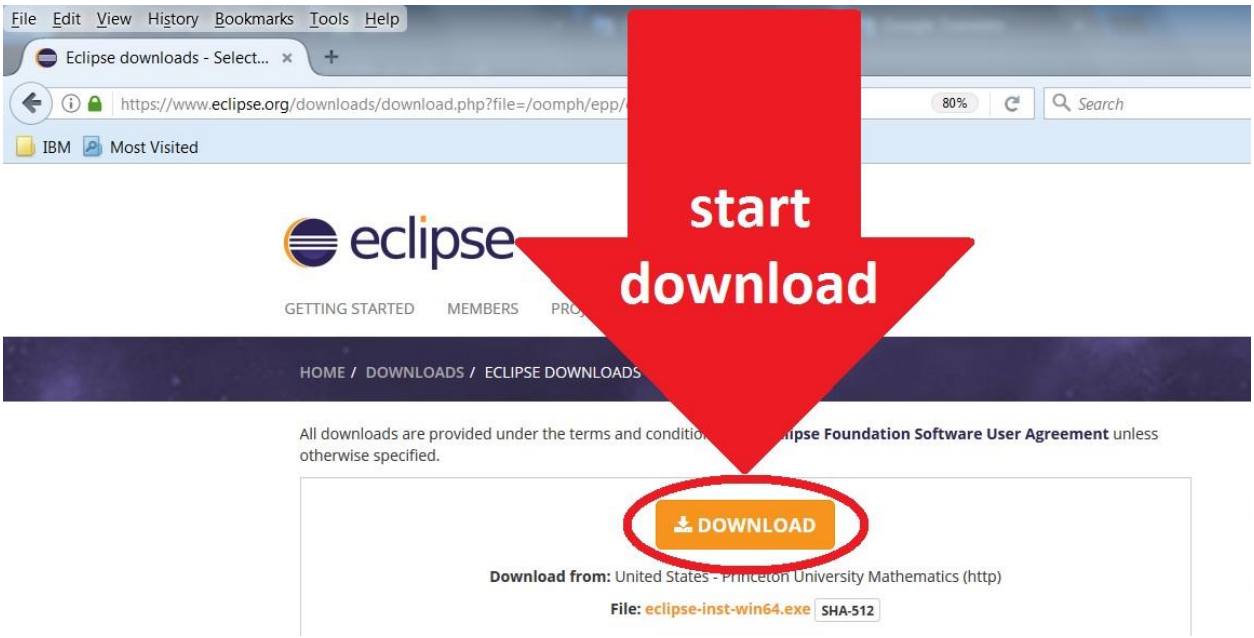


Attention, tomcat must not be installed, just unzip it and remember the folder path.

3. Eclipse

Download and install Eclipse Oxygen from this address:

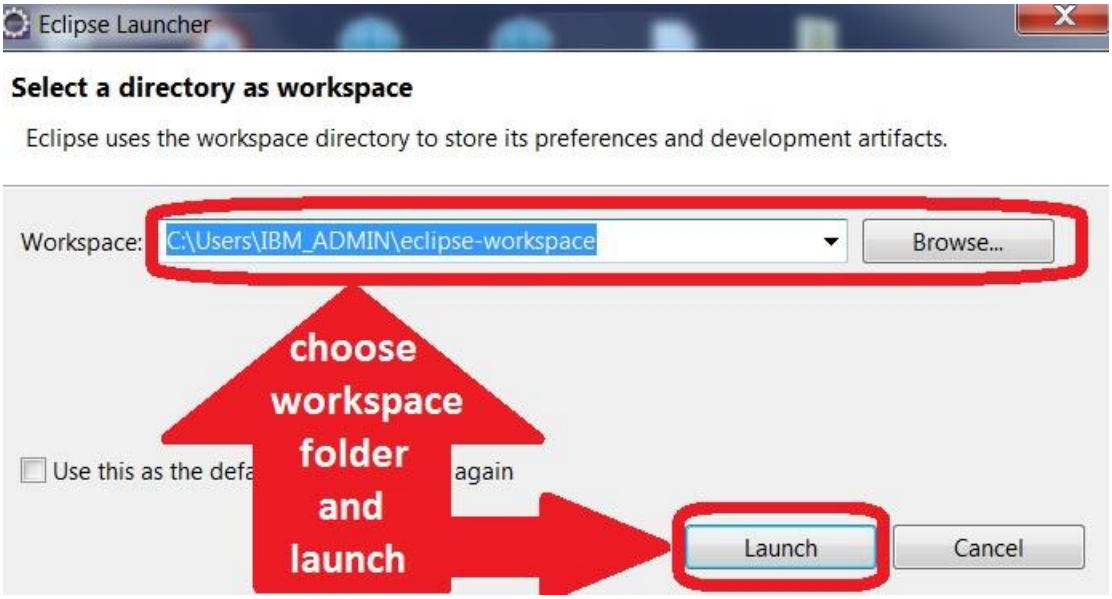
https://www.eclipse.org/downloads/download.php?file=/oomph/epp/oxygen/R/eclipse-inst-win64.exe



On installation process, choose Eclipse IDE for Java EE Developers:

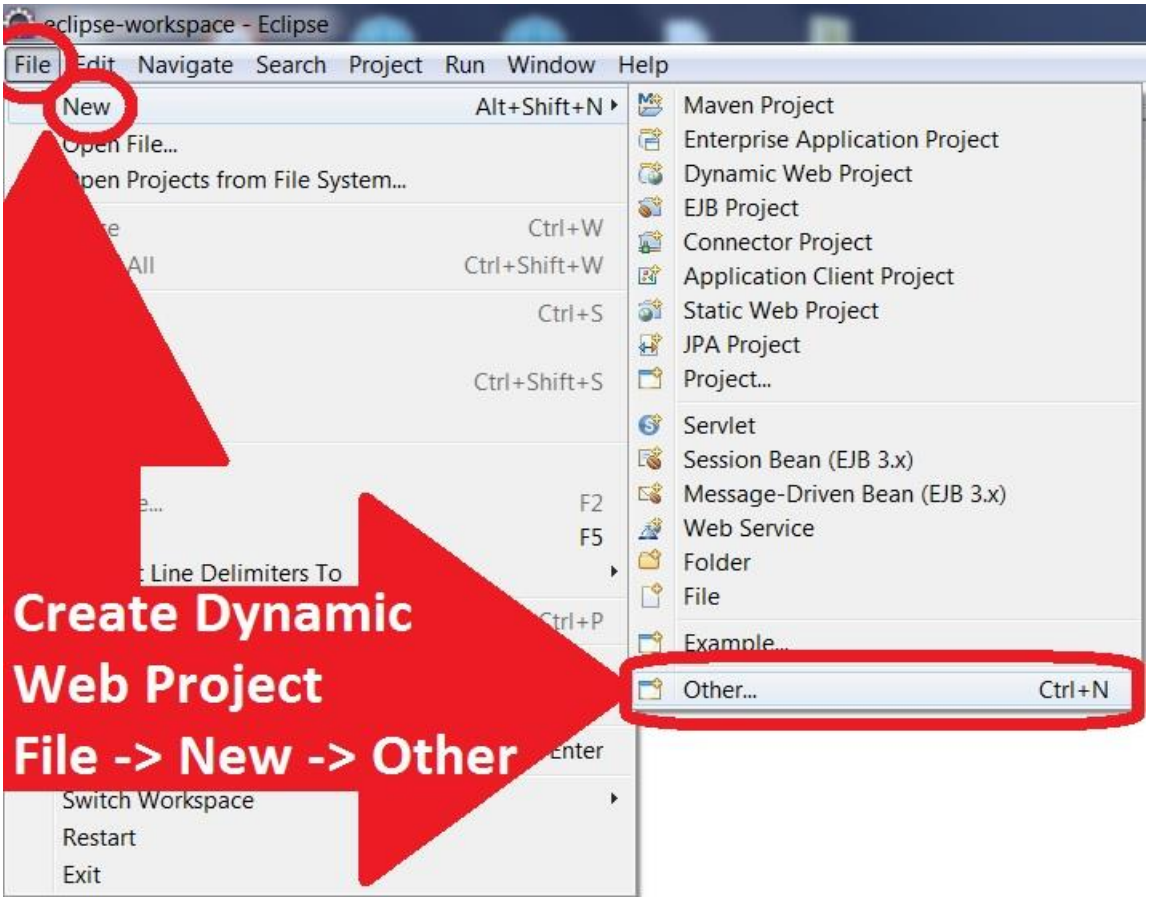
Launch eclipse:

Select proper workspace:

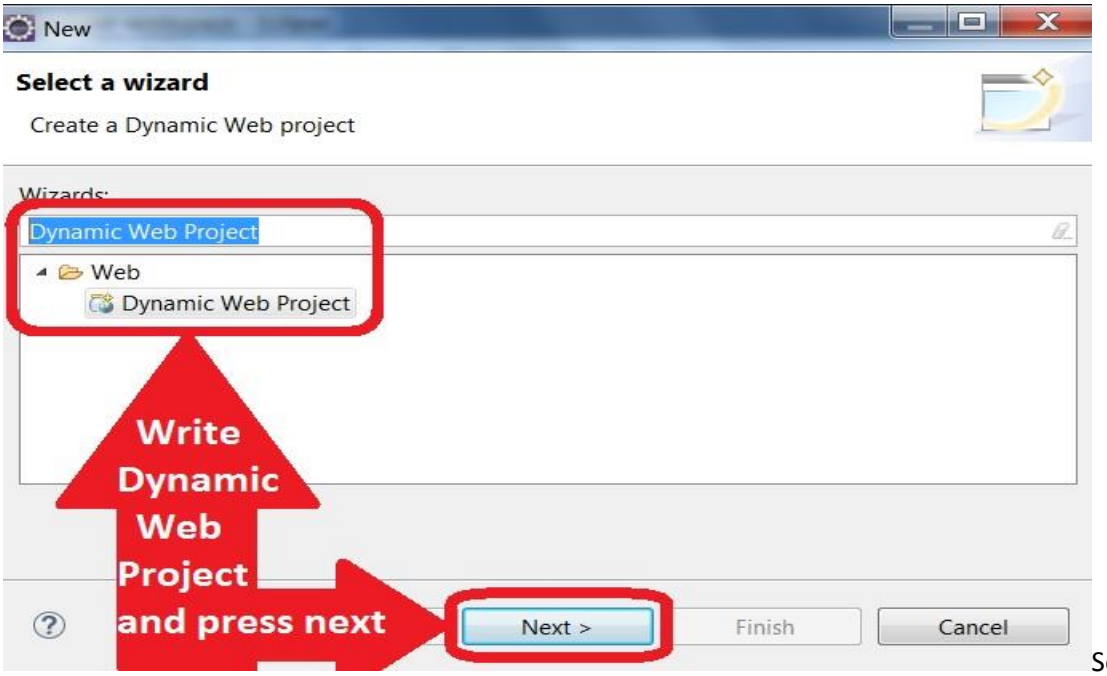


4. Install Spring application

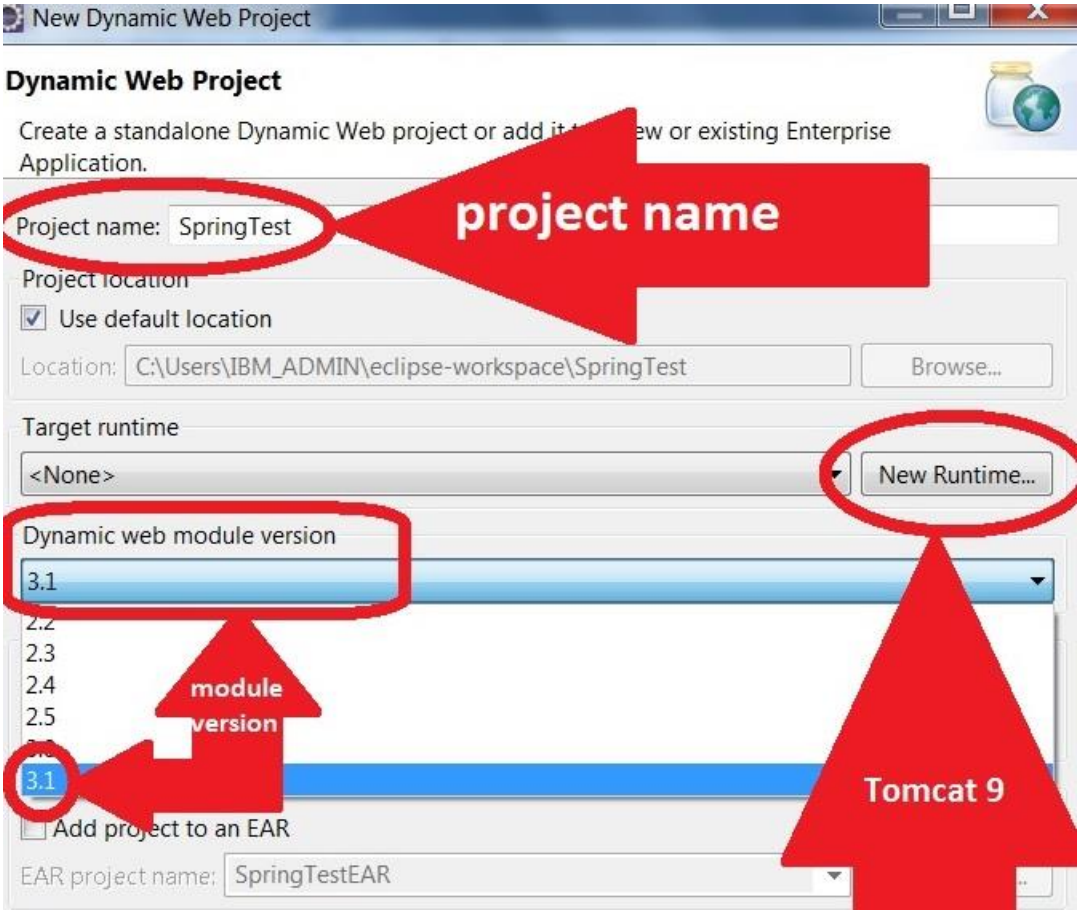
In eclipse, create Dynamic Web Project. **File -> New -> Other**:



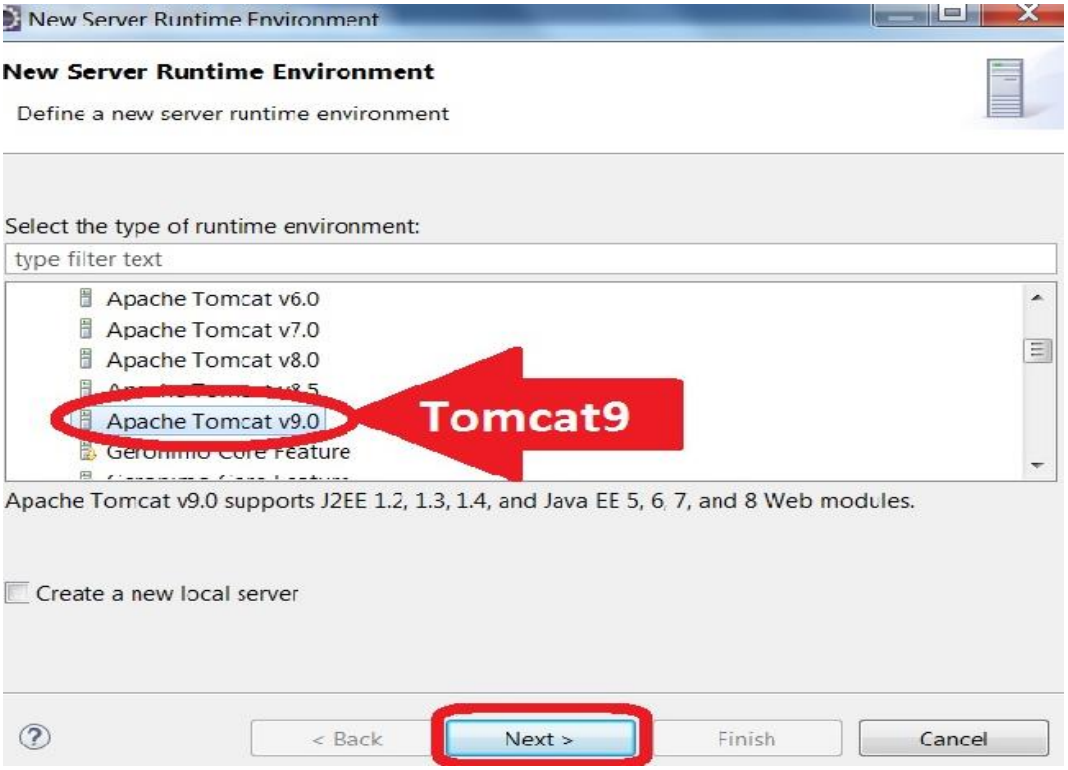
Write **Dynamic Web Project** and press next:



Set name, Project name, tomcat version and module version:

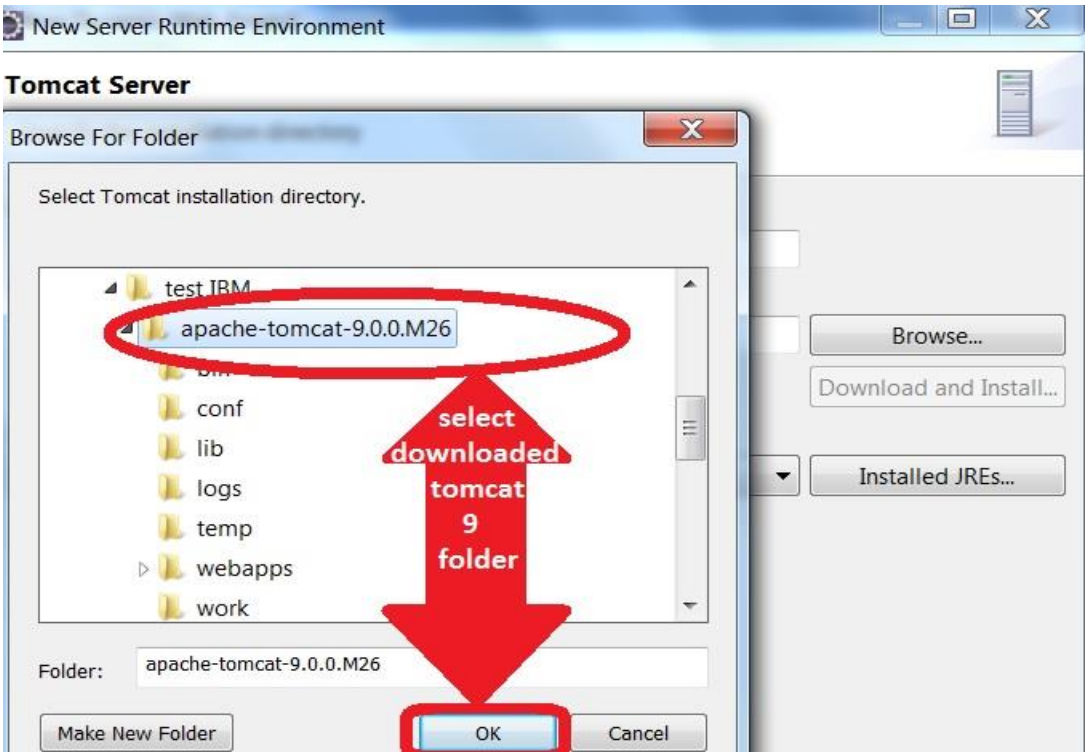


Choose Tomcat 9:

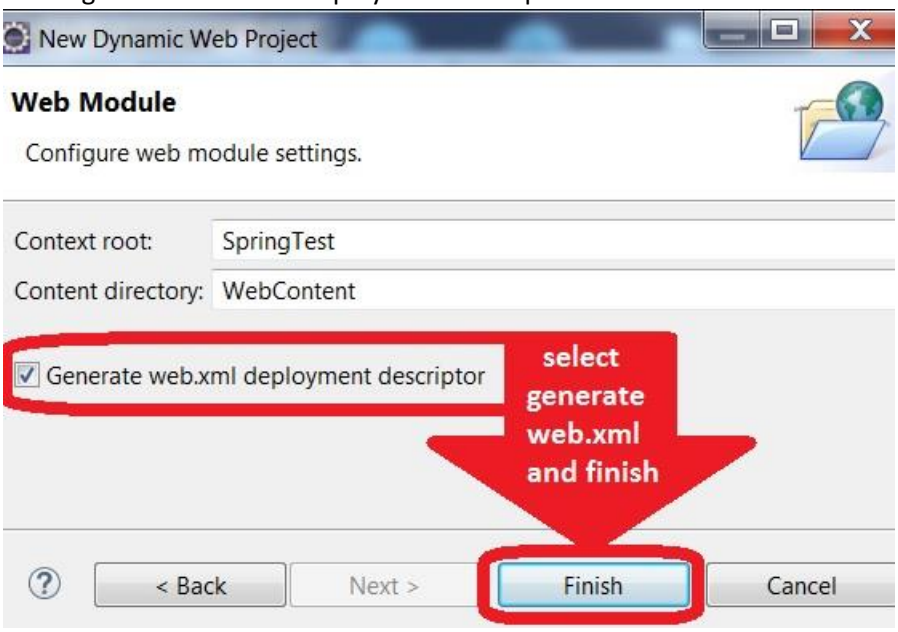


Set Tomcat 9 and JDK:

Select downloaded tomcat 9 folder:



Select generate web.xml deployment descriptor and finish:

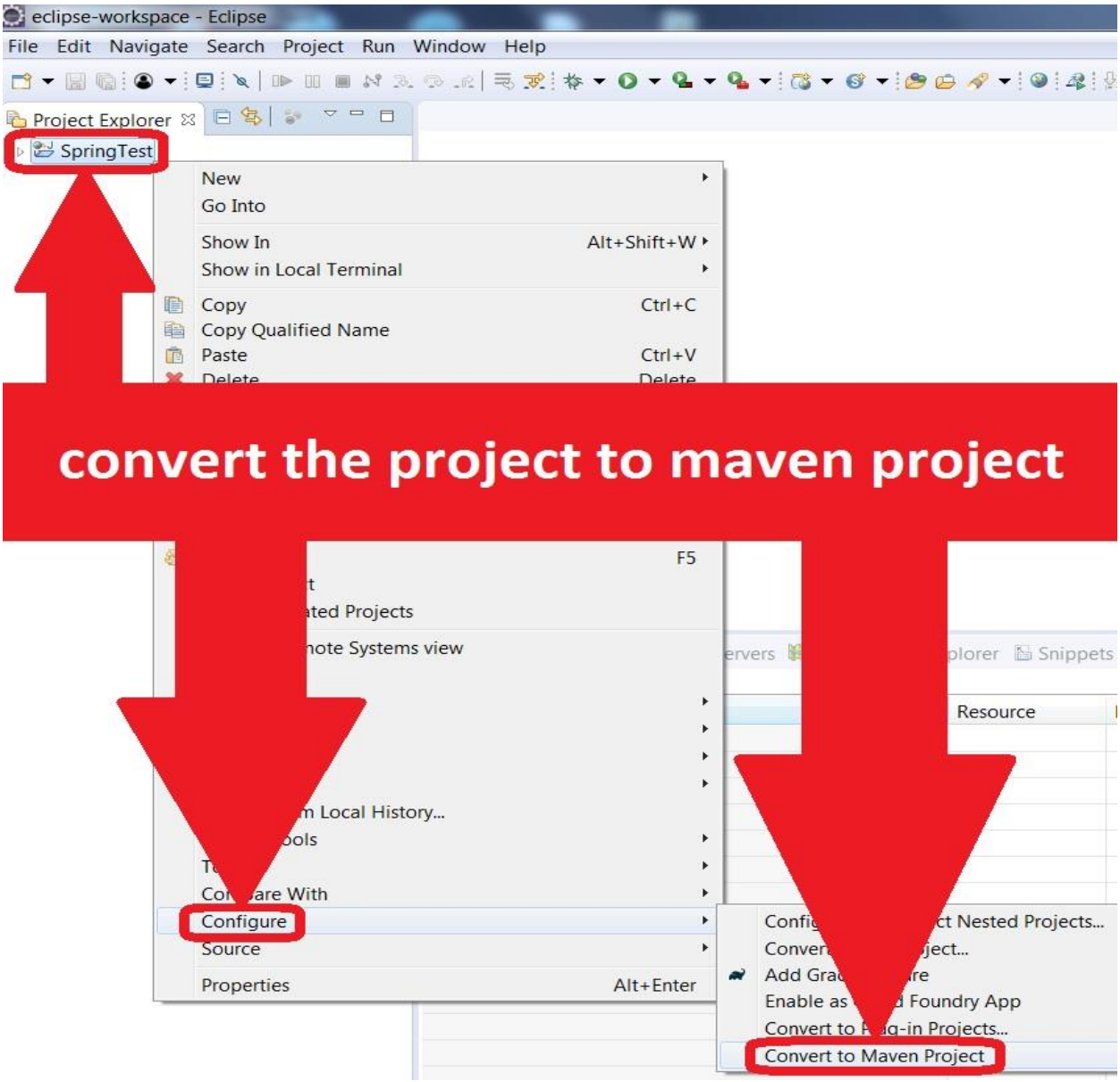


5. Maven

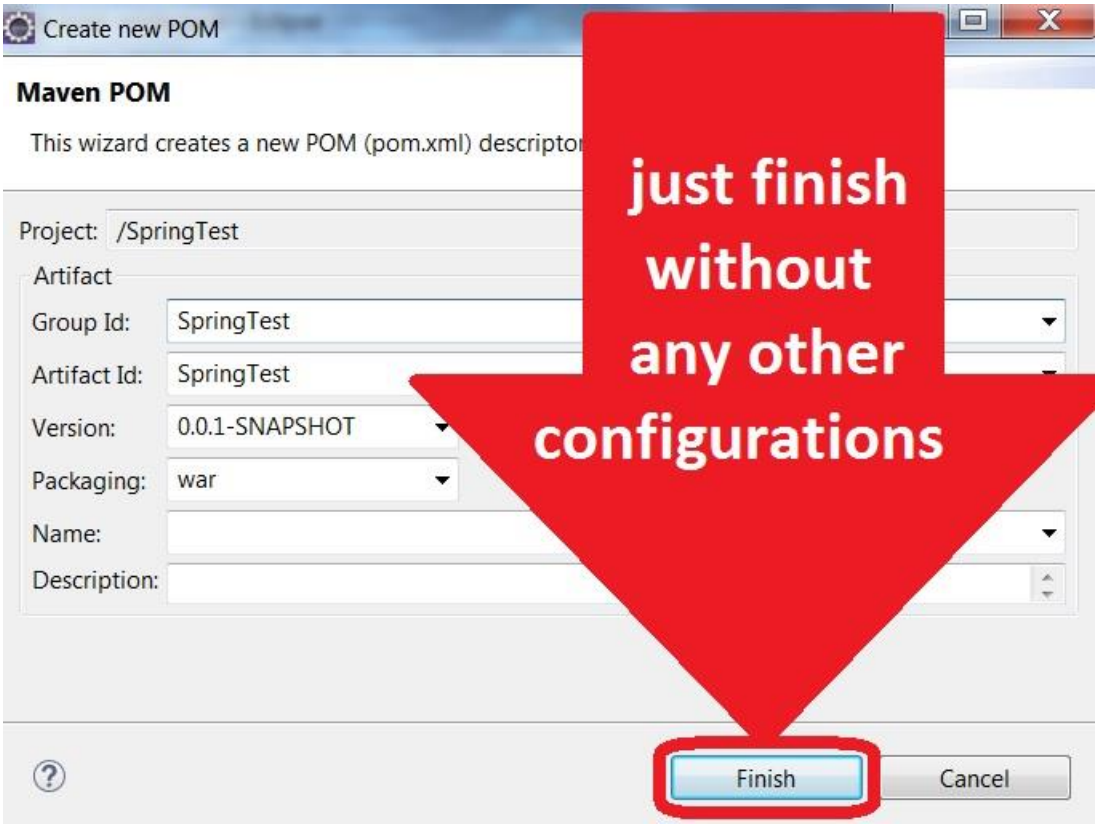
In order to be able to use different java libraries, you need to use maven. So, use eclipse to convert your project to maven project like in next picture.

As an alternative, you can add these libraries manually. So, you can avoid maven, but will be more difficult to maintenance, update etc.

If you decide to use maven: Right click on project -> **Configure -> Convert to maven project**:



Finish maven conversion:



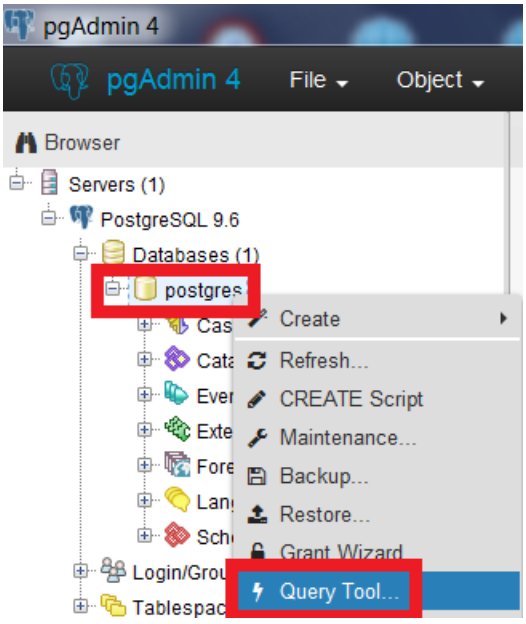
6. Source code

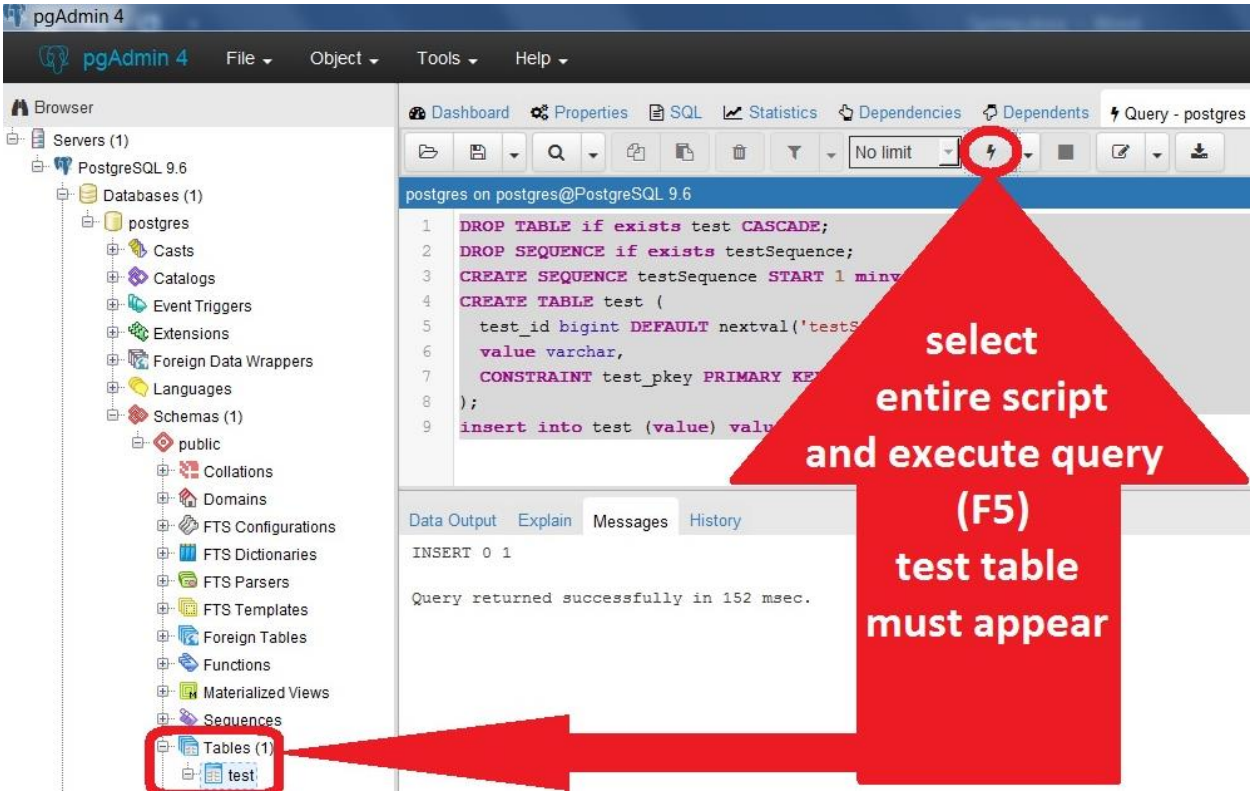
Until now you successfully created an empty project. From now, you must start implement spring MVC code application.

In order to create a database test table, run this script into pgAdmin tool:

```
DROP TABLE if exists test CASCADE;

DROP SEQUENCE if exists testSequence;

CREATE SEQUENCE testSequence START 1 minvalue 0;

CREATE TABLE test (

 test_id bigint DEFAULT nextval('testSequence') NOT NULL,

 value varchar,

 CONSTRAINT test_pkey PRIMARY KEY(test_id)

);

insert into test (value) values('database value');
```

If you not find Query Tool, make right click on postgres and choose Query Tool:



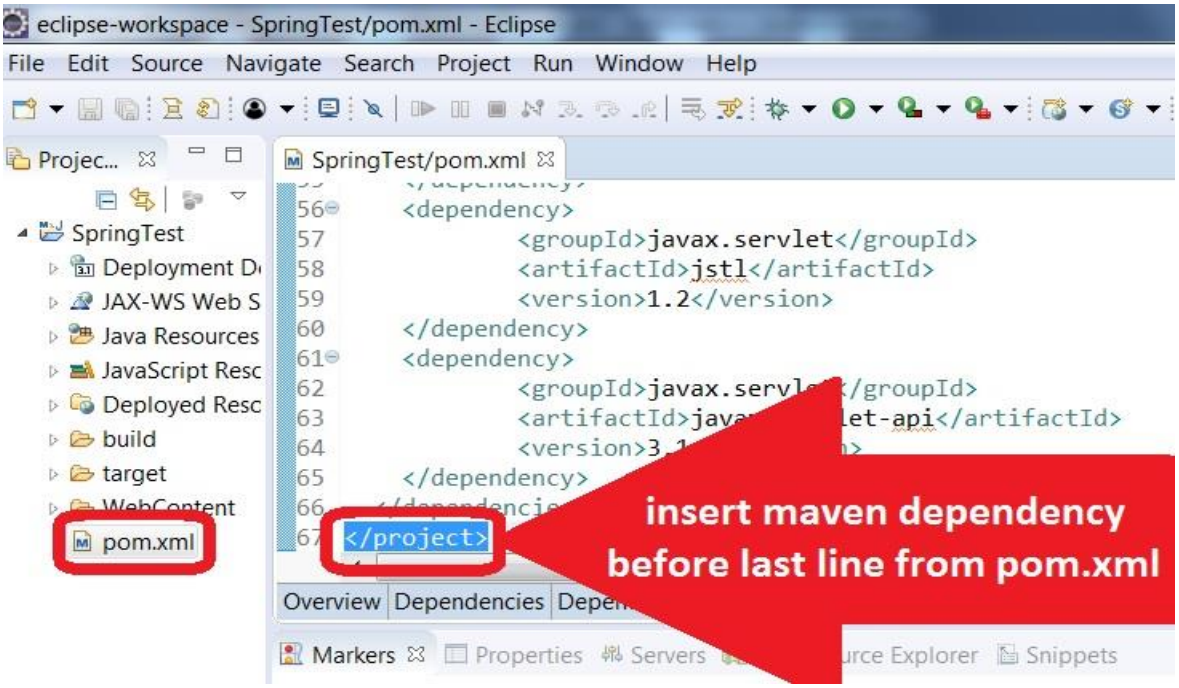


pom.xml

Go into your pom.xml file and insert maven dependencies, before last line "</project>":

```
<properties>
    <spring.version>4.3.10.RELEASE</spring.version>
</properties>
<dependencies>
        <dependency>
                    <groupId>org.springframework</groupId>
                    <artifactId>spring-webmvc</artifactId>
                    <version>${spring.version}</version>
        </dependency>
        <dependency>
        <groupId>org.springframework</groupId>
            <artifactId>spring-orm</artifactId>
```

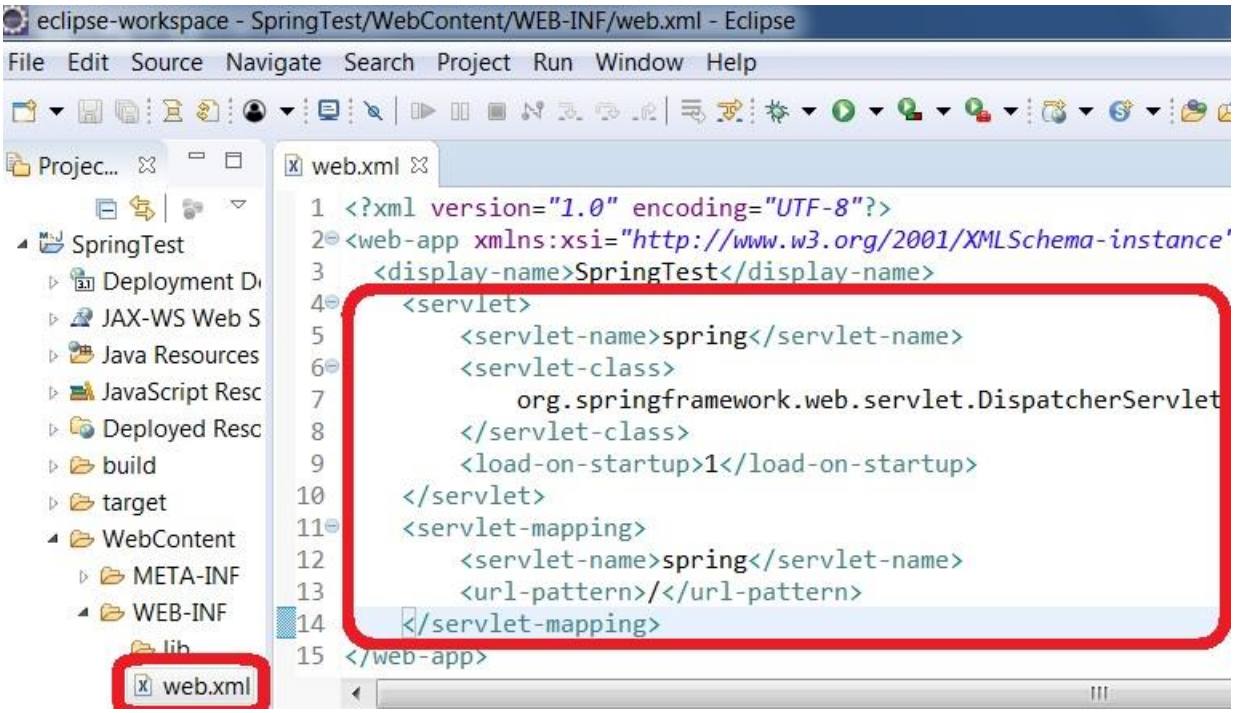
```
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>9.1-901-1.jdbc4</version>
        </dependency>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.2.10.Final</version>
        </dependency>
        <dependency>
            <groupId>jstl</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
        </dependency>
    </dependencies>
```



web.xml

In web.xml file, delete **</welcome-file-list>** tag and his contain and replace with:

```
<servlet>
        <servlet-name>spring</servlet-name>
        <servlet-class>
                org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>spring</servlet-name>
        <url-pattern>/</url-pattern>
</servlet-mapping>
```



spring-servlet.xml

Create a new xml file named "spring-servlet.xml" and add it into **WebContent -> WEB-INF** with the following source code:

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"

xmlns:mvc="http://www.springframework.org/schema/mvc"
xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-4.0.xsd
http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">

<context:component-scan base-package="controller" />

<context:component-scan base-package="dao" />

<context:component-scan base-package="config" />

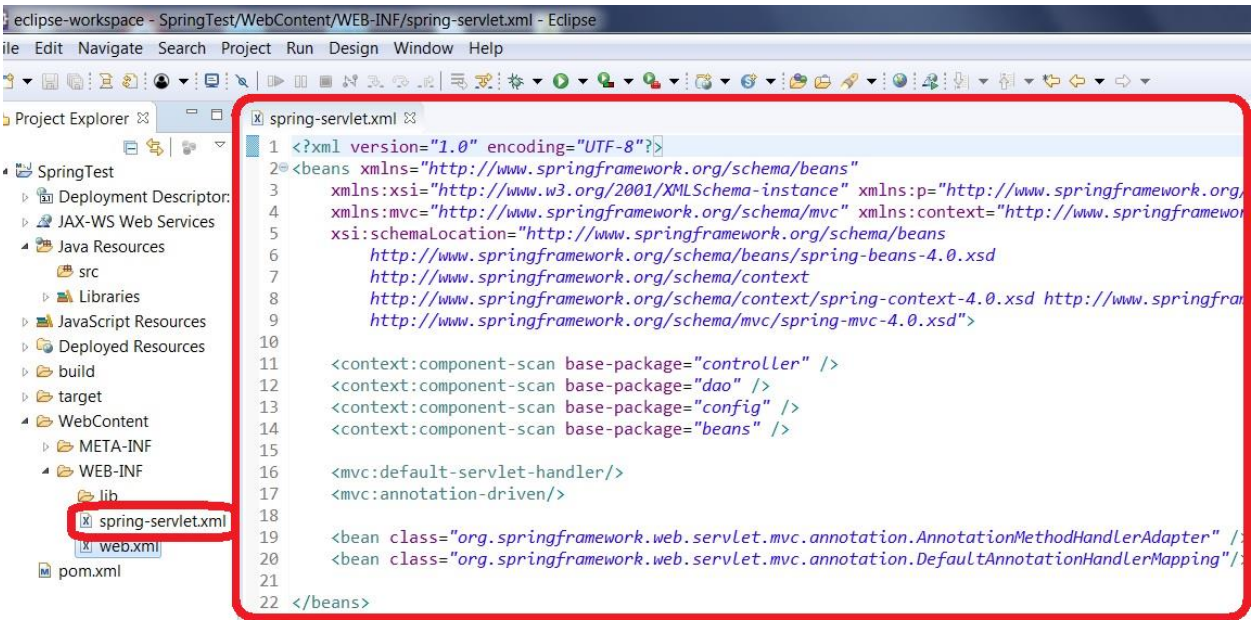
<context:component-scan base-package="beans" />

<mvc:default-servlet-handler/>

<mvc:annotation-driven/>

<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter" />

<bean class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping"/>
</beans>



JSP view files

Create a folder named "**jsp**" into **WebContent -> WEB-INF**. Here, create two jsp files **page_1.jsp** and **page_2.jsp**.

page_1.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>

<html>

<head>


<meta http-equiv="Content-Type" content="text/html; charset=utf-8">


<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>


<style>


div.tab {

   overflow: hidden;

   background-color: #ffffff;

}


div.tab button {

  background-color: inherit;

  float: left;

  border: none;

  outline: none;

  cursor: pointer;

  font-size: 17px;
```

```
  border-radius: 55px;
  padding: 10px;
  width:100px;
  height: 40px;
}

div.tab button:hover {
   background-color: #ddd;
}

div.tab button.active {
   background-color: #39D951;
}
</style>

</head>
<body>

        <c:if test="${not empty page_1}">

                <div class="tab">
                        <button class="active">Page 1</button>
                        <button id="page_2_Id">Page 2</button>
                </div>
            <br>
                <form:form action="page_1" modelAttribute="page_1" id="page_1_form">

                                <form:input path="eventId"
```

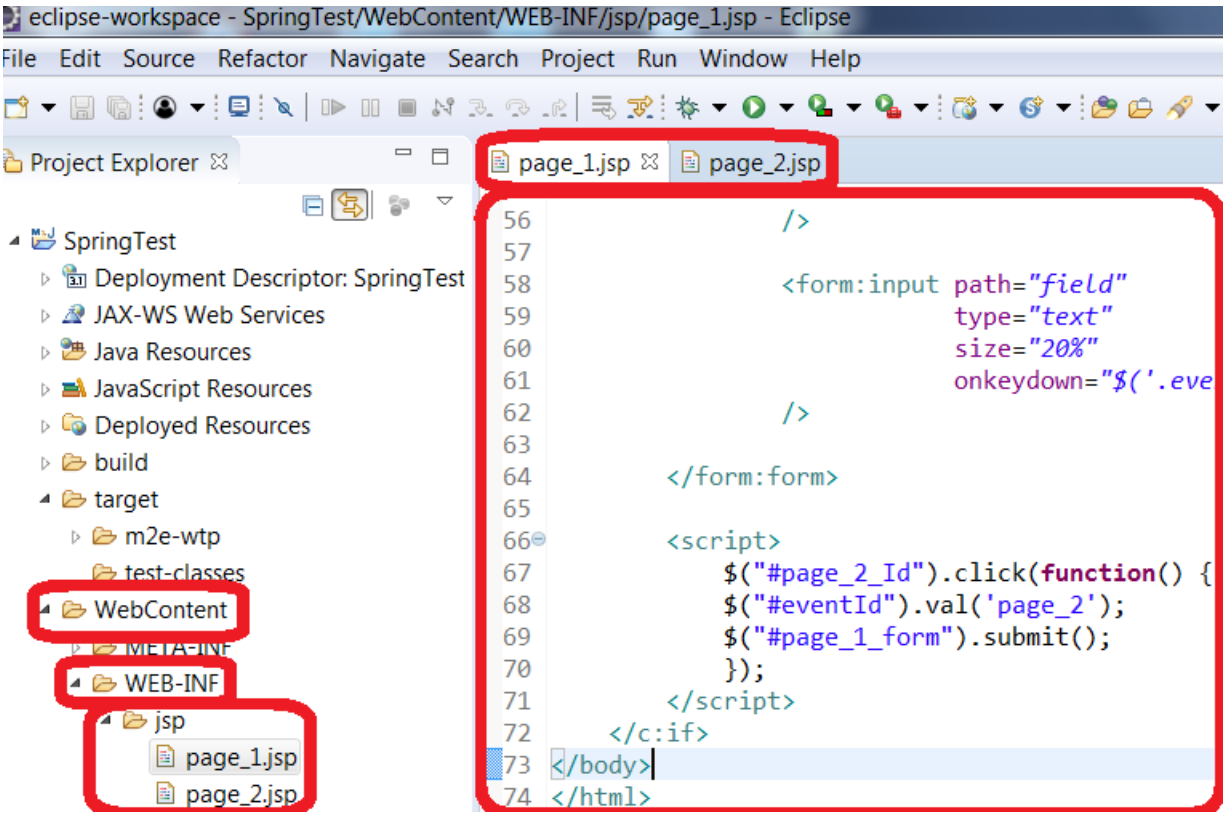
```
                                        type="text"
                                        class="eventId"
                                        hidden="true"
                                />

                                <form:input path="field"
                                        type="text"
                                        size="20%"
        onkeydown="$('.eventId').val('page_1');if(event.keyCode==13){this.form.submit();};"
                                />

                </form:form>

                <script>
                        $("#page_2_Id").click(function() {
                        $("#eventId").val('page_2');
                        $("#page_1_form").submit();
                        });
                </script>
        </c:if>
</body>
</html>
```

page_2.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<html>
<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>

<style>

div.tab {
    overflow: hidden;
    background-color: #ffffff;
}

div.tab button {
   background-color: inherit;
   float: left;
   border: none;
   outline: none;
   cursor: pointer;
   font-size: 17px;
   border-radius: 55px;
   padding: 10px;
```

```
  width:100px;
  height: 40px;
}

div.tab button:hover {
   background-color: #ddd;
}

div.tab button.active {
   background-color: #39D951;
}
</style>

</head>
<body>

        <c:if test="${not empty page_2}">

                <div class="tab">
                        <button id="page_1_Id">Page 1</button>
                        <button class="active">Page 2</button>
                </div>

                <form:form action="page_2" modelAttribute="page_2" id="page_2_form">

                        <form:input path="eventId"
                                                type="text"
                                                class="eventId"
```

```
                                        hidden="true"
                />


        </form:form>


        <script>
                $("#page_1_Id").click(function() {
                $("#eventId").val('page_1');
                $("#page_2_form").submit();
                });
        </script>


    </c:if>

</body>
</html>
```



Into **Java Resources ->src** create these packages:

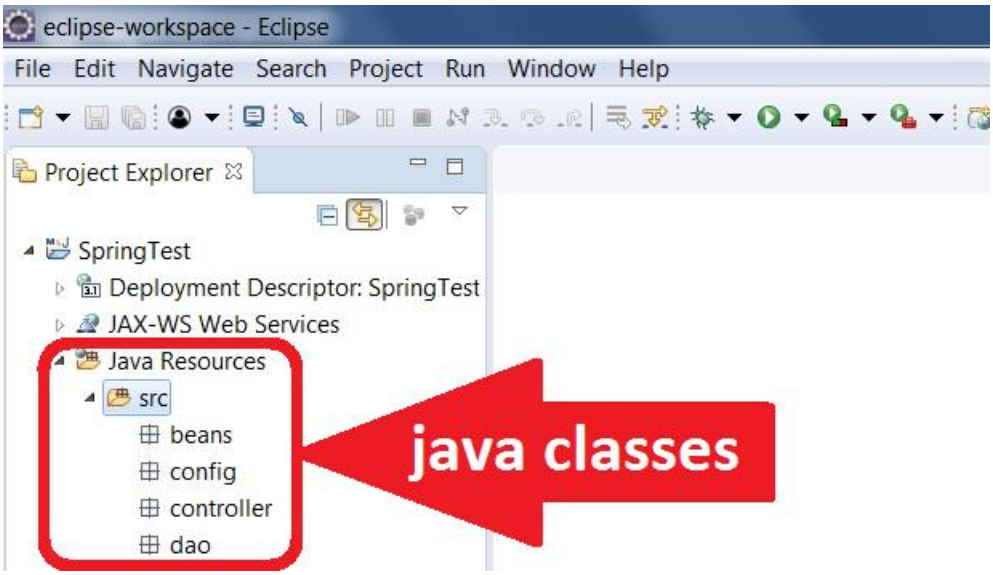
beans

config

controller

dao

Even if the packages appear outside the src folder, just select src folder and make refresh(F5).



Create the following java classes.

- into **beans**: **Page_1.java**, **Page_2.java**, **TestTable.java**
- into **config**: **ApplicationContextConfig.java**
- into **controller**: **FirstController.java**, **Page_1_Controller.java**, **Page_2_Controller.java**
- into **dao**: **DAO.java(is an interface, not a class)**, **DAOImpl.java**

Page_1.java:

```
package beans;


public final class Page_1 {


        private String eventId;

        private String field;


        public String getEventId() {

                return eventId;

        }

        public void setEventId(String eventId) {

                this.eventId = eventId;
```

```
	}
	public String getField() {
		return field;
	}
	public void setField(String field) {
		this.field = field;
	}
}
```

Page_2.java:

```
package beans;

public class Page_2 {

	private String eventId;

	public String getEventId() {
		return eventId;
	}

	public void setEventId(String eventId) {
		this.eventId = eventId;
	}
}
```

TestTable.java:

```
package beans;
```

```
import java.io.Serializable;


import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;


import org.springframework.context.annotation.Scope;
import org.springframework.context.annotation.ScopedProxyMode;
import org.springframework.stereotype.Component;


@Component
@Entity(name = "test")
@Table(name = "test")
@Scope(proxyMode = ScopedProxyMode.TARGET_CLASS, value = "session")
public class TestTable implements Serializable {


        private static final long serialVersionUID = -1776173986706192926L;


        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Basic(optional = false)
        @Column(name = "test_id", nullable = false)
        private long test_id;
```

```
	@Column(name = "value")
	private String value;

	public long getTest_id() {
		return test_id;
	}

	public void setTest_id(long test_id) {
		this.test_id = test_id;
	}

	public String getValue() {
		return value;
	}

	public void setValue(String value) {
		this.value = value;
	}
}
```

ApplicationContextConfig.java:

```
package config;

import javax.sql.DataSource;

import org.hibernate.SessionFactory;
import org.springframework.context.annotation.Bean;
```

```
import org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBuilder;
import org.springframework.transaction.annotation.EnableTransactionManagement;
import org.springframework.web.servlet.ViewResolver;
import org.springframework.web.servlet.config.annotation.DefaultServletHandlerConfigurer;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;

import beans.TestTable;

@EnableWebMvc
@Configuration
@EnableTransactionManagement
public class ApplicationContextConfig extends WebMvcConfigurerAdapter{

	@Override
	public void configureDefaultServletHandling(DefaultServletHandlerConfigurer configurer) {
		configurer.enable();
	}

	@Bean
	public ViewResolver getViewResolver() {
		InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
		viewResolver.setPrefix("/WEB-INF/jsp/");
```

```
        viewResolver.setSuffix(".jsp");
        viewResolver.setViewClass(JstlView.class);
        return viewResolver;
    }

    @Bean
    public DataSource getDataSource() {

        DriverManagerDataSource dataSource = new DriverManagerDataSource();

        dataSource.setDriverClassName("org.postgresql.Driver");
        dataSource.setUrl("jdbc:postgresql://localhost:5432");
        dataSource.setUsername("postgres"); // your username
        dataSource.setPassword("ibm");      // your password
        return dataSource;
    }

    @Bean
    public SessionFactory getSessionFactory(DataSource dataSource) {

        LocalSessionFactoryBuilder sessionBuilder = new
LocalSessionFactoryBuilder(dataSource);

        sessionBuilder.addAnnotatedClasses(TestTable.class);

        return sessionBuilder.buildSessionFactory();
    }
```

```
	@Bean(name = "transactionManager")
	public HibernateTransactionManager getTransactionManager(SessionFactory sessionFactory)
{

		return new HibernateTransactionManager(sessionFactory);
	}
}
```

In order to set a database connection, username and password must be from your previous postgres installation. Different credentials will make connection not available.

dataSource.setUsername("your username ");

dataSource.setPassword("your password");

FirstController.java:

```
package controller;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;


import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.servlet.ModelAndView;


import beans.Page_1;

import beans.Page_2;

import beans.TestTable;

import dao.DAO;


@Controller
```

```
public class FirstController {

	@Autowired
	private DAO dao;

	@RequestMapping(value = "/")
	public ModelAndView first(ModelMap model, HttpServletRequest request){

		try{

			if(request != null){

				HttpSession session = request.getSession();

				if(session != null){

					final Page_1 page_1 = new Page_1();
					final Page_2 page_2 = new Page_2();

					if(dao != null){
						TestTable dbTest = dao.getTestById(1);

						if(dbTest != null){
							page_1.setField(dbTest.getValue());
						}
					}

					session.setAttribute("page_1",page_1);
```

```
                                        session.setAttribute("page_2",page_2);


                                        model.addAttribute("page_1", page_1);
                                }
                        }
                }catch(Exception e){
                        e.printStackTrace();
                }

                return new ModelAndView("page_1", model);
        }
}
```

Page_1_Controller.java:

```
package controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import beans.Page_1;
import beans.Page_2;
```

```
import beans.TestTable;

import dao.DAO;


@Controller

public class Page_1_Controller {


	@Autowired

	private DAO dao;


	@RequestMapping(value = "/page_1", method = RequestMethod.POST)

	public ModelAndView test(Page_1 page_1_Form, ModelMap model, HttpServletRequest request){


		try{


			if(request != null){


				HttpSession session = request.getSession();


				if(session != null){


					Page_1 page_1_Bean = (Page_1)session.getAttribute("page_1");


					if(page_1_Bean != null){


						String fieldValue = page_1_Form.getField();


						if(dao != null){
```

```
                                        TestTable dbTest = new TestTable();
                                        dbTest.setValue(fieldValue);

                                        boolean isSaved =
dao.isSaveOrUpdateTest(dbTest);

                                        page_1_Bean.setField(fieldValue + " " +
isSaved);

                                }
                        }

                        String eventId = page_1_Form.getEventId();

                        if(eventId != null){

                                if(eventId.equals("page_1")){

                                        model.addAttribute("page_1", page_1_Bean);

                                        return new ModelAndView("page_1", model);
                                }
                                else if(eventId.equals("page_2")){

                                        Page_2 page_2_Bean =
(Page_2)session.getAttribute("page_2");
                                        model.addAttribute("page_2", page_2_Bean);
```

```
                                                return new ModelAndView("page_2", model);
                                            }
                                        }
                                    }
                                }
                            }
                            catch(Exception e){
                                e.printStackTrace();
                            }
                            return null;
                    }
}
```

Page_2_Controller.java:

```
package controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

import beans.Page_1;
import beans.Page_2;
```

```
@Controller
public class Page_2_Controller {

	@RequestMapping(value = "/page_2", method = RequestMethod.POST)
	public ModelAndView test(Page_2 page_2_Form, ModelMap model, HttpServletRequest
request){

		try{

			if(request != null){

				HttpSession session = request.getSession();

				if(session != null){

					String eventId = page_2_Form.getEventId();

					if(eventId != null){

						if(eventId.equals("page_1")){

							Page_1 page_1_Bean =
(Page_1)session.getAttribute("page_1");

							model.addAttribute("page_1", page_1_Bean);

							return new ModelAndView("page_1", model);
						}
						else if(eventId.equals("page_2")){
```

```
                                                Page_2 page_2_Bean =
(Page_2)session.getAttribute("page_2");
                                                model.addAttribute("page_2", page_2_Bean);

                                                return new ModelAndView("page_2", model);
                                        }
                                    }
                                }
                            }
                        }
                        catch(Exception e){
                                e.printStackTrace();
                        }
                        return null;
                }
}
```

DAO.java:

```
package dao;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import beans.TestTable;

@Repository("dao")
@Transactional
public interface DAO {
```

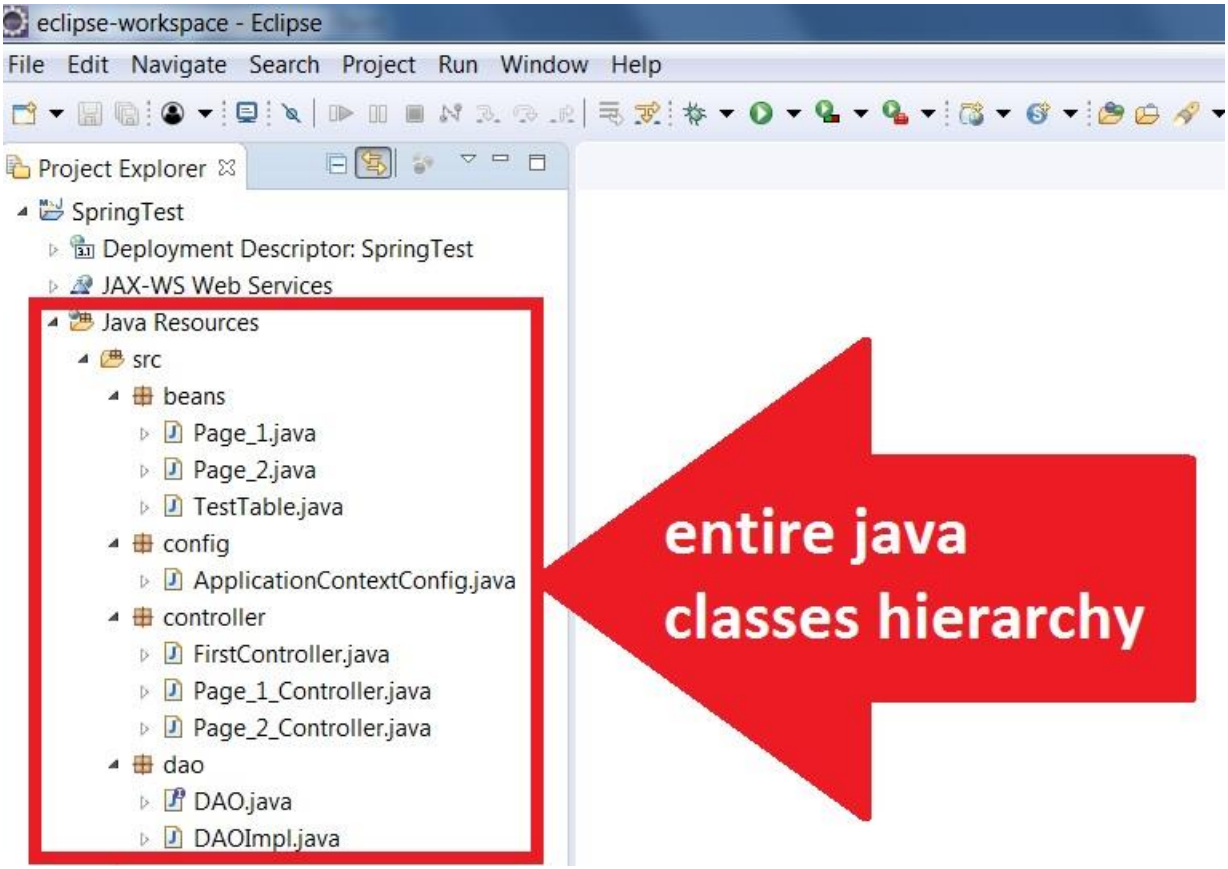
```
	public TestTable getTestById(long id);
	public boolean isSaveOrUpdateTest(TestTable test);
}
```

DAOImpl.java:

```
package dao;
import java.util.List;

import org.hibernate.SessionFactory;
import org.hibernate.query.Query;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.transaction.annotation.Transactional;

import beans.TestTable;

@Repository("dao")
@Transactional
public class DAOImpl implements DAO {

	@Autowired
	private SessionFactory sessionFactory;

	@SuppressWarnings("unchecked")
	@Override
	@Transactional
	public TestTable getTestById(long id){

		try{
```

```
                String userQuery = "from " + TestTable.class.getName() + " where test_id =
:test_id ";

                Query<TestTable> query =
sessionFactory.getCurrentSession().createQuery(userQuery);

                query.setParameter("test_id", id);

                List<TestTable> testList = query.getResultList();

                if(testList != null){
                        for(TestTable test : testList){
                                if(test != null){
                                        return test;
                                }
                        }
                }
        }
        catch(Exception e){
                e.printStackTrace();
        }
        return null;
    }

    @Override
    @Transactional
    public boolean isSaveOrUpdateTest(TestTable test){
```

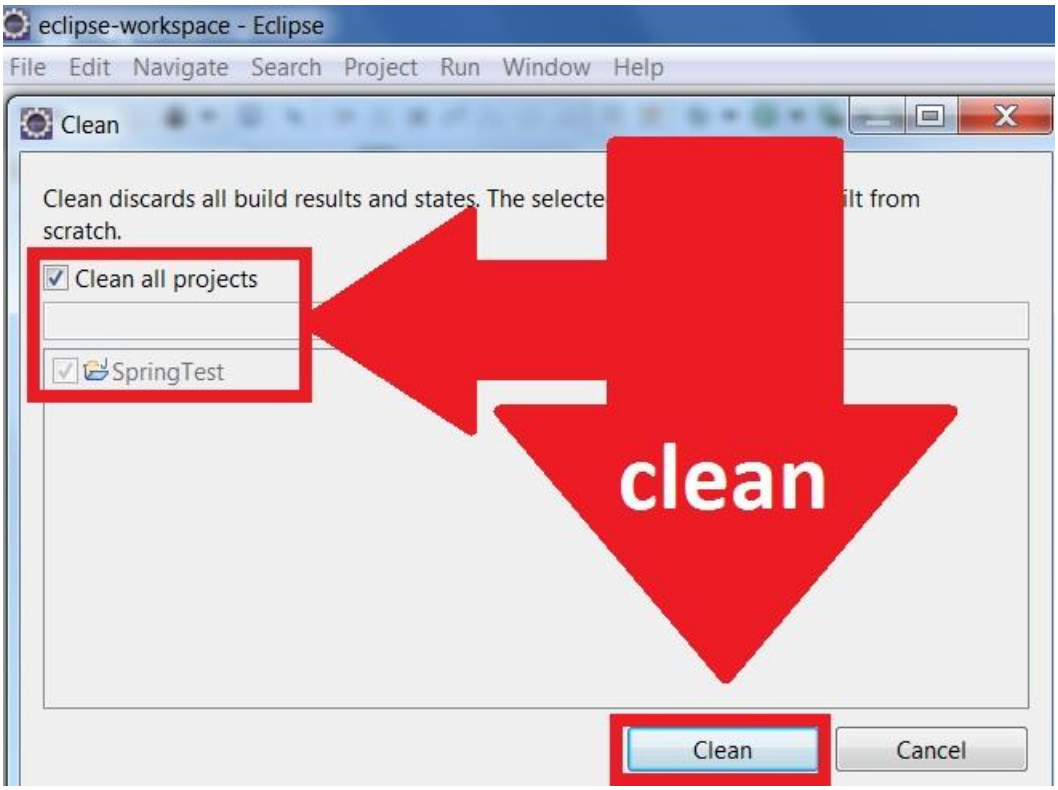
```
            try{
                if(test != null){
                    sessionFactory.getCurrentSession().saveOrUpdate(test);
                    return true;
                }
            }
            catch(Exception e){
                e.printStackTrace();
                return false;
            }
            return false;
        }
}
```



7. Run application

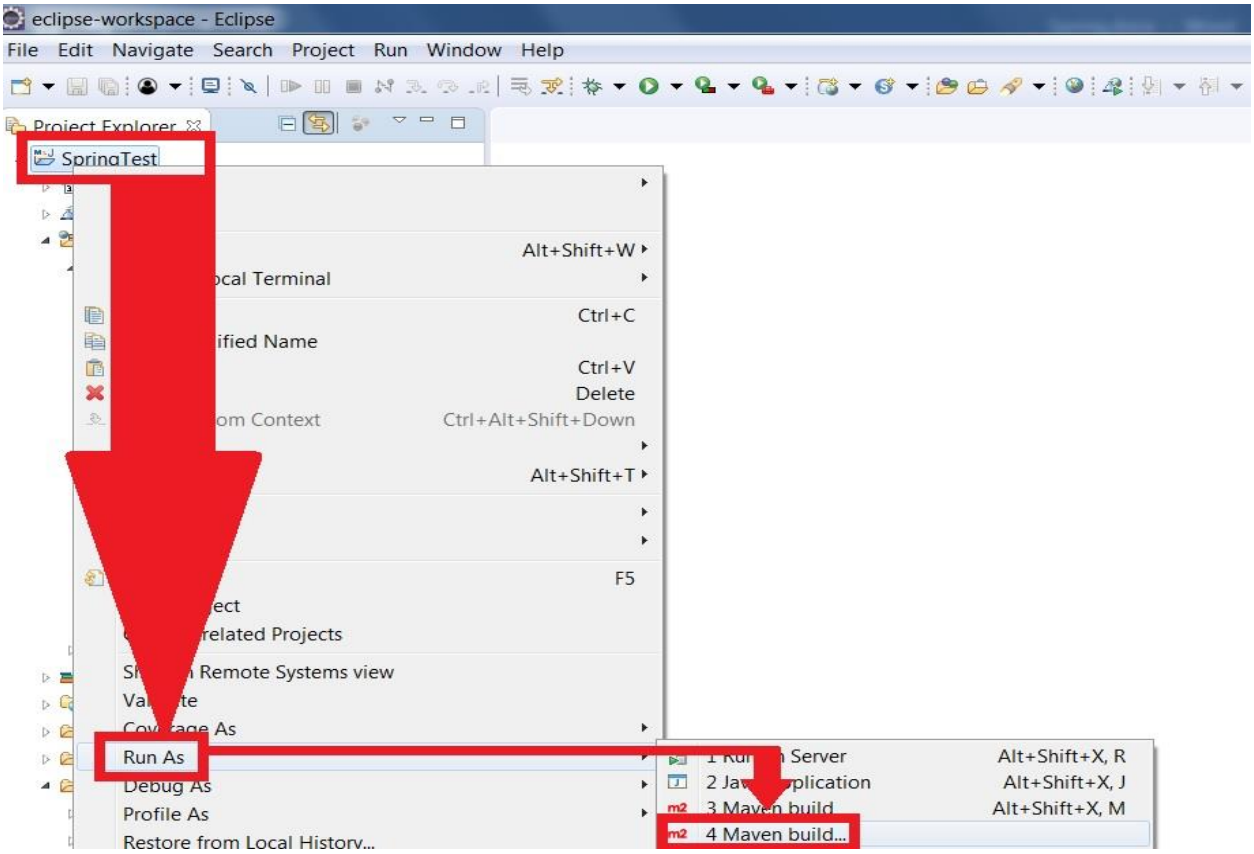
Before run the application, you must do these steps: clean, build(clean install), update maven, refresh. Don't forget to set JDK on your project, otherwise the build will not run.

Clean, **Project->Clean**:

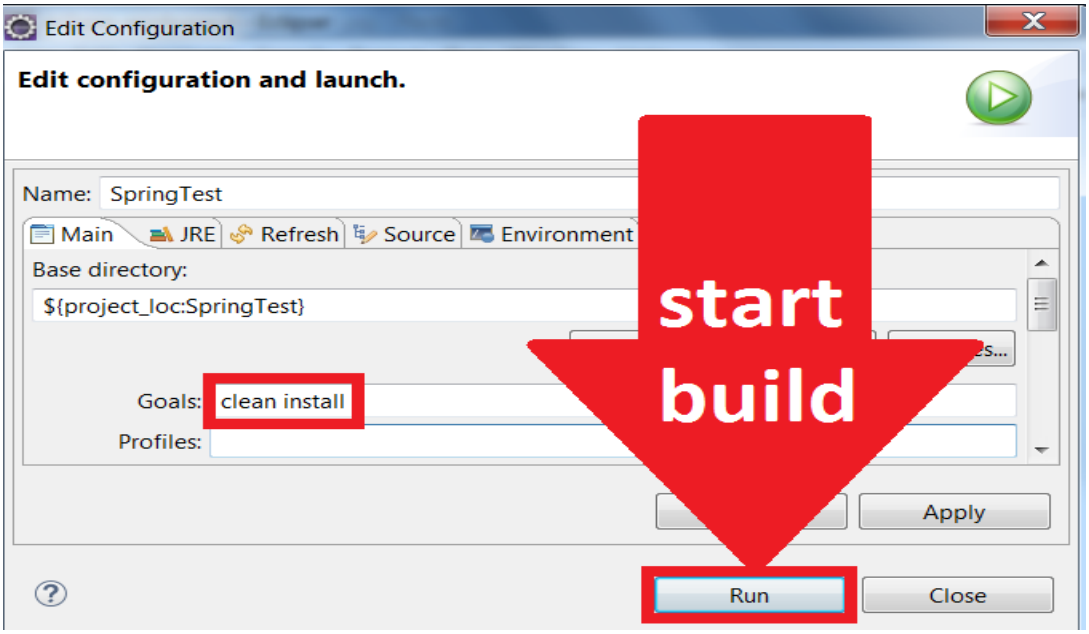


Build

Right click on **SpringTest -> Run As -> Maven Build**

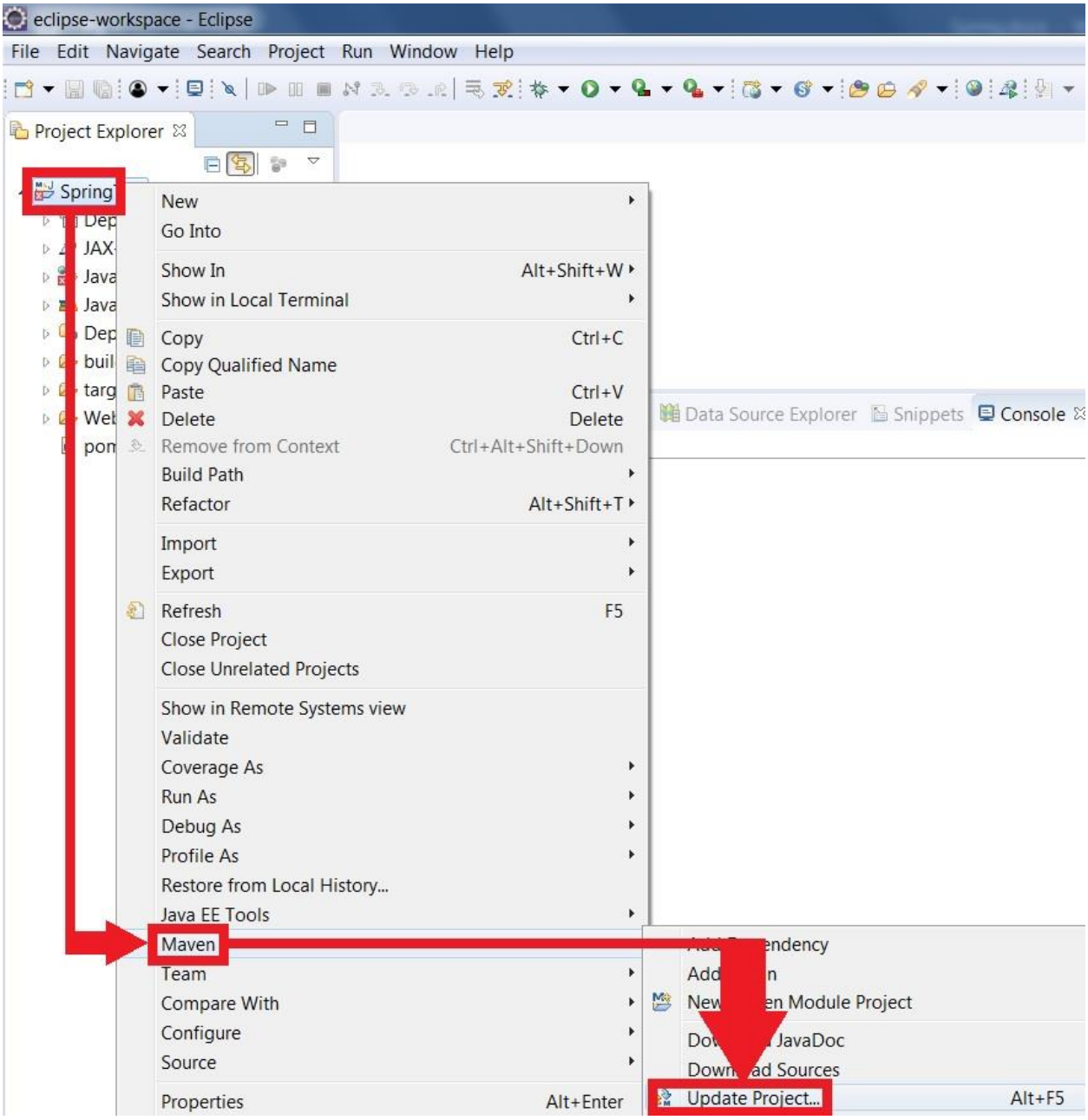


Type **clean install** and run the build:

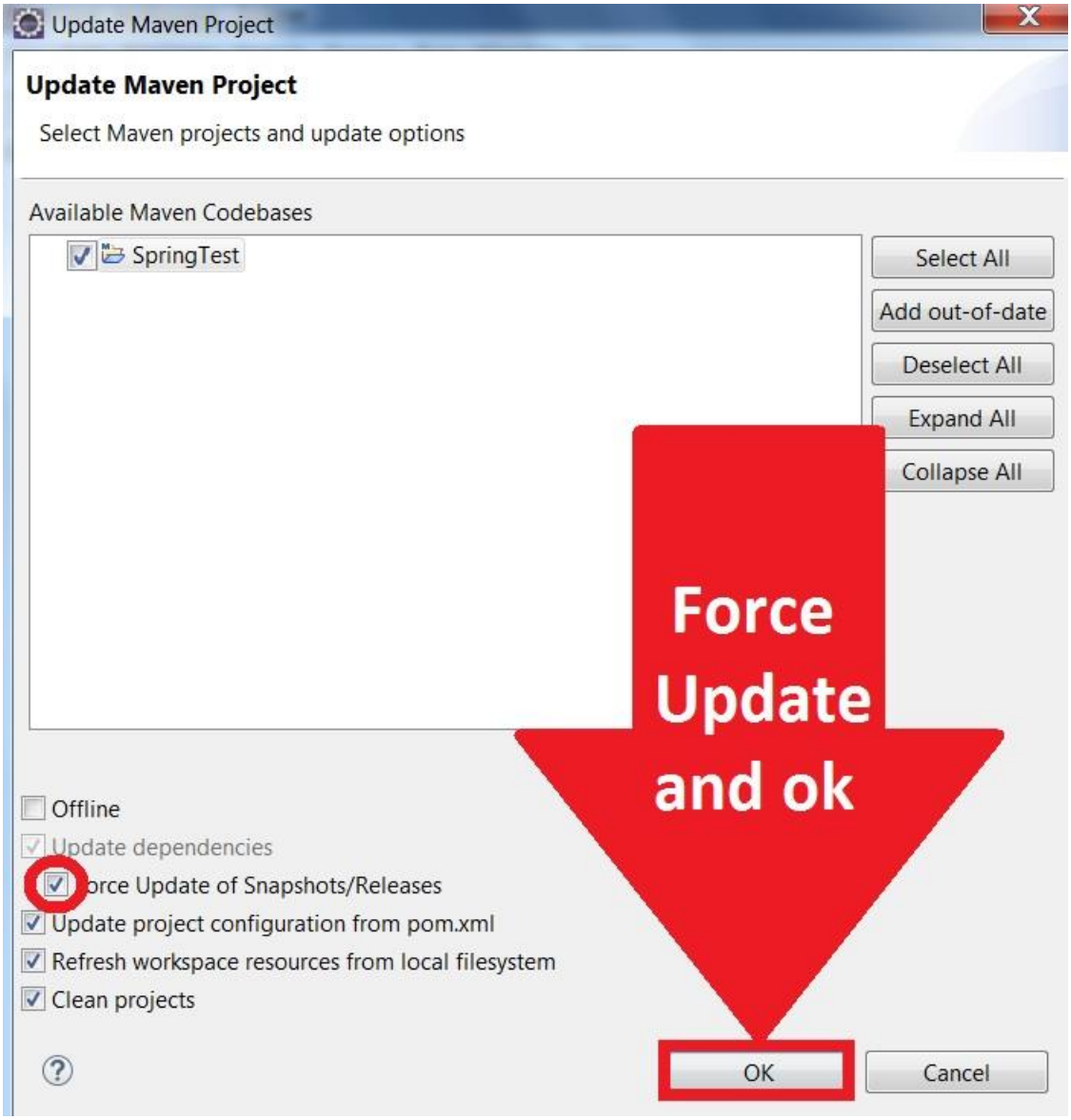


Update maven.

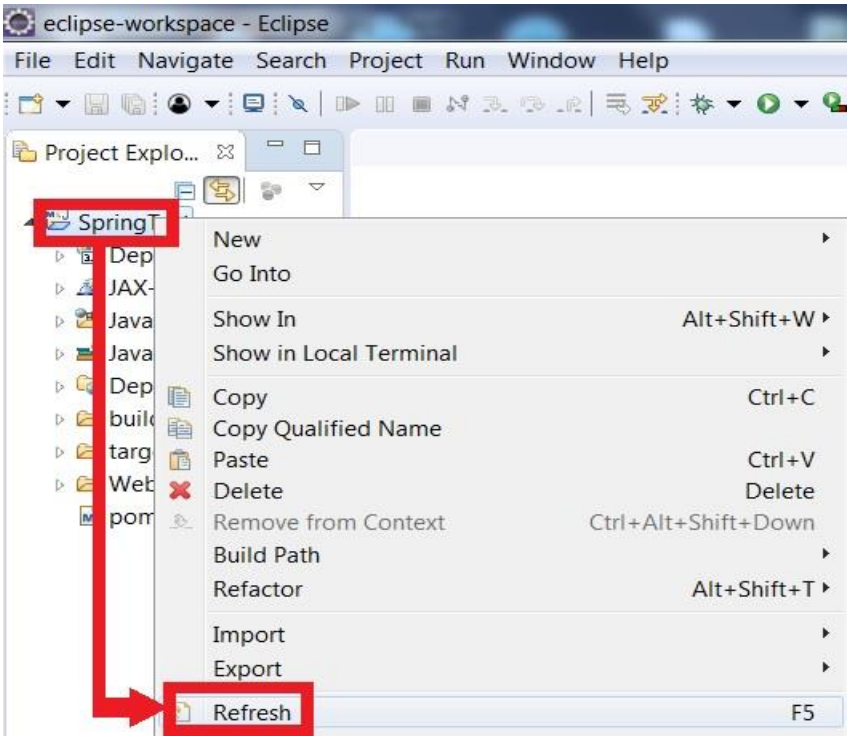
Right click on **SpringTest -> Maven -> Update Project**



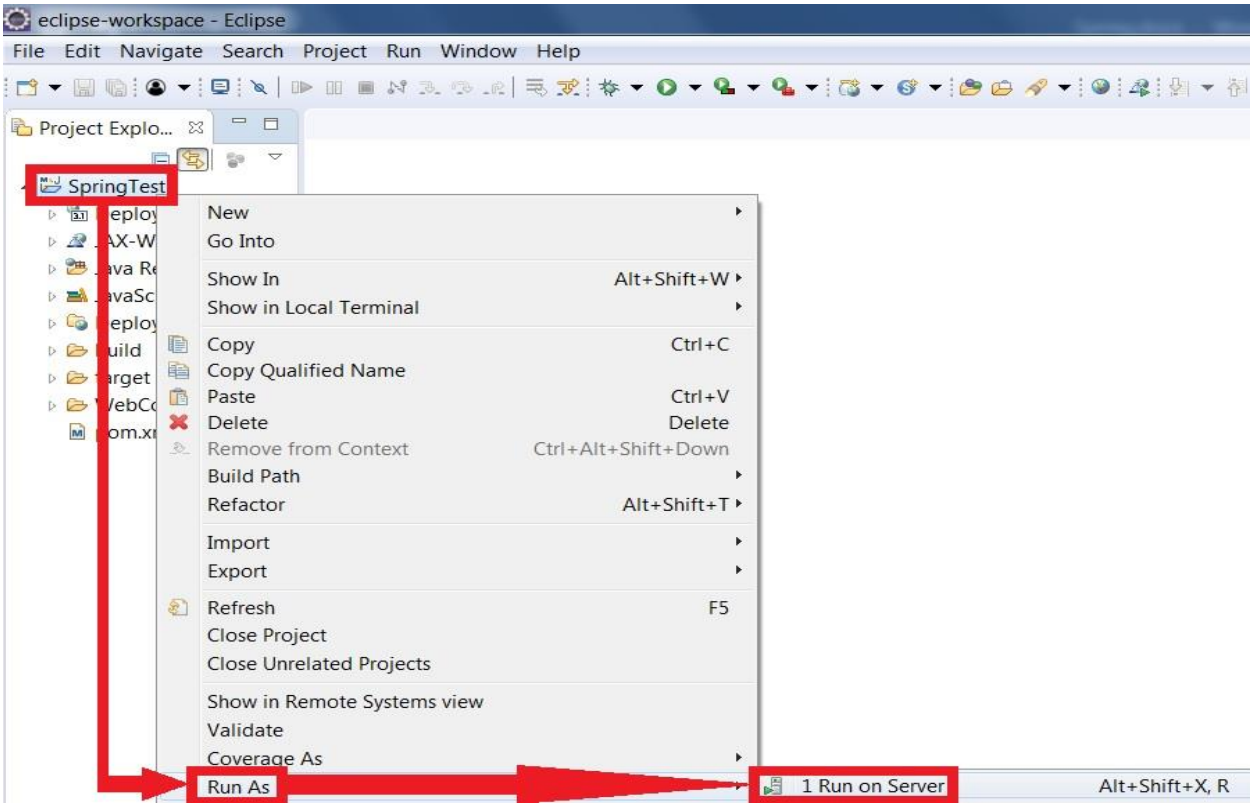
Start update maven



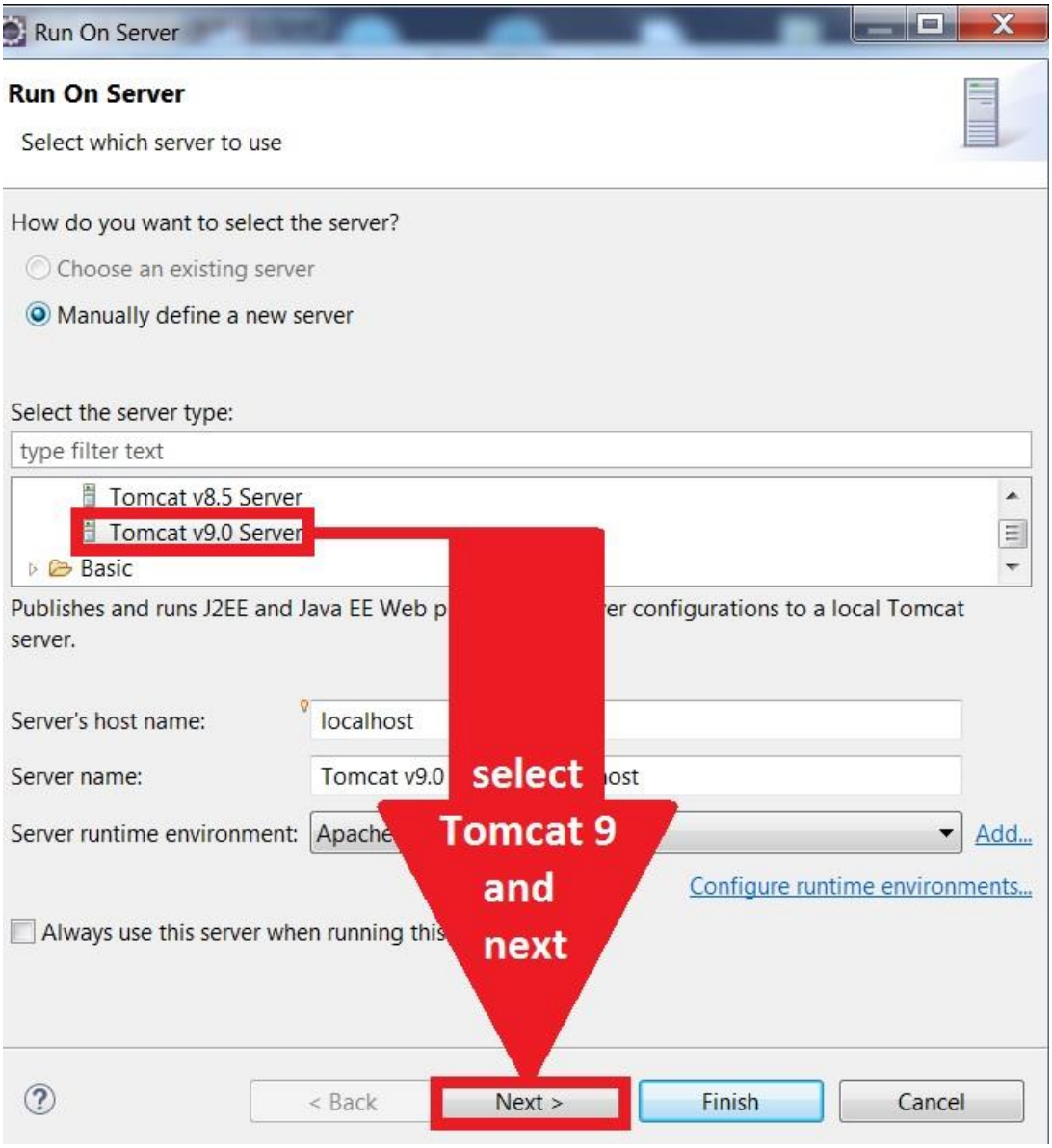
Refresh:



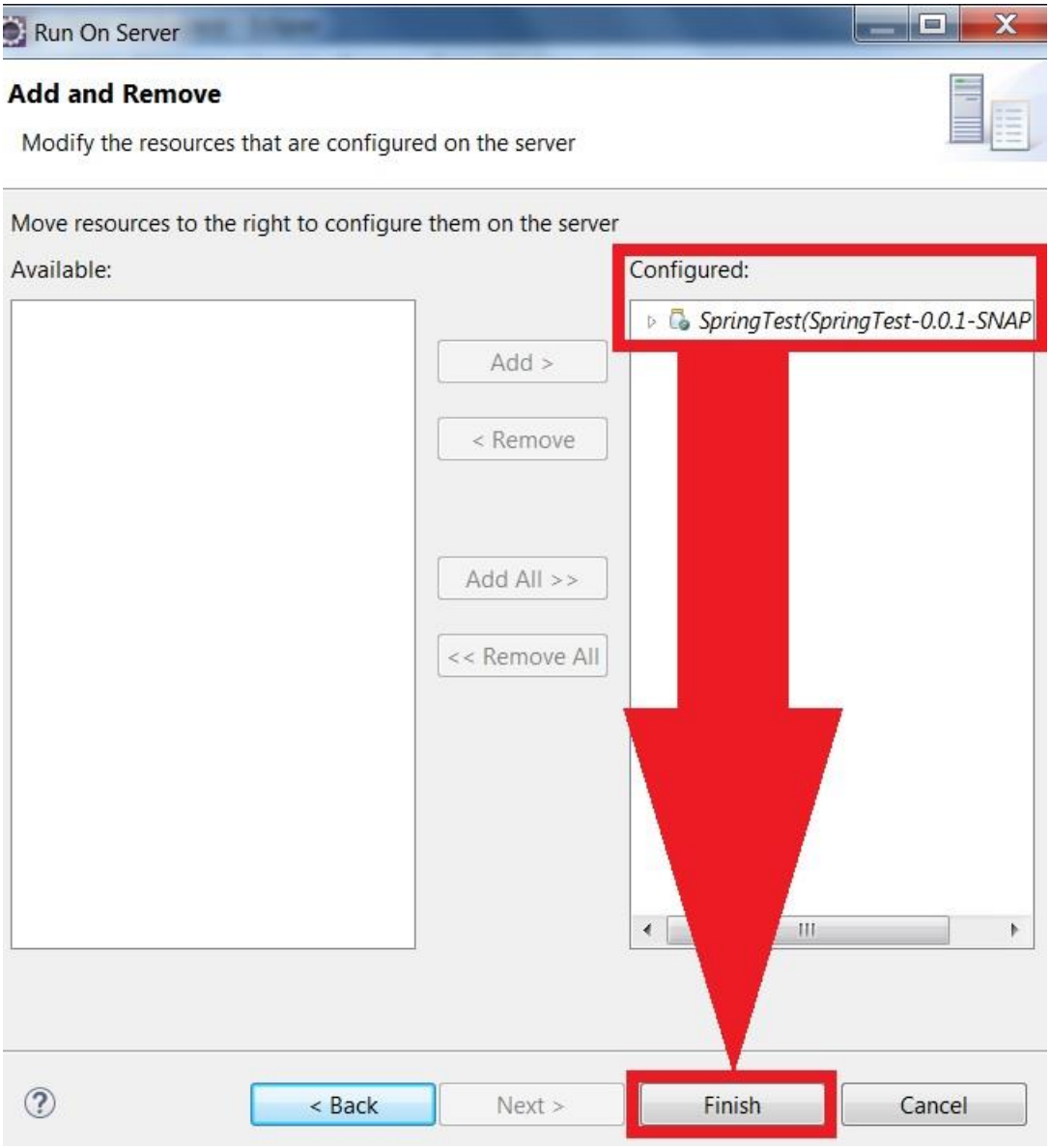
Right click on **SpringTest -> Run As -> Run on Server**



Select Tomcat version:



Configure project and finish



The demo test application should start: